Available online at www.sciencedirect.com
ScienceDirect
Computers and Mathematics with Applications 54 (2007) 282–295
An International Journal
computers & mathematics with applications
www.elsevier.com/locate/camwa

# A new approximation algorithm for obtaining the probability distribution function for project completion time

Ming-Jong Yao*, Weng-Ming Chu

*Department of Industrial Engineering and Enterprise Information, Tunghai University, P.O. Box 985, Taichung City, 407, Taiwan*

Received 20 January 2006; received in revised form 9 April 2006; accepted 22 January 2007

**Abstract**

This paper focuses on the application of the techniques of discretization to obtain an approximated probability density function (*pdf*) for the completion time of large-size projects, in which we allow any type of *pdf* for the duration of activities. In this study, we improve the techniques of discretization in the following two ways: first, we propose to replace the max operation with an approximation procedure to save significant computational loading; and second, to reduce the error from assuming independence between paths using a simple heuristic rule. To evaluate the performance of our proposed algorithm, we randomly generated 20 sets of 100-node instances in our numerical experiments. Taking the results from a Monte Carlo simulation using 20,000 samples as a benchmark, we demonstrate that the proposed algorithm significantly outperforms the PERT model and Dodin's [B.M. Dodin, Approximating the distribution function in stochastic networks, Comput. Oper. Res. 12 (3) (1985) 251–264] algorithm in both the running time and the precision aspects.


*Keywords:* Stochastic activity networks; Completion time; Probability distribution function; Approximation; Discretization

## 1. Introduction

There is evidence (see [2] and [3]) which indicates that for large projects, only approximately 10%∼15% of them finish on time. The majority of them overrun not only their planned schedule, but also their project budget. The problem of determining *the completion time of a project* has been extensively dealt with in management science/operations research/industrial engineering. The educators and the researchers in these areas advocate representing a project as a *network* and applying the solution approaches for network models to them to assist project managers to monitor their completion. (It explains why the mathematical models for project management are known together as the *theory of activity networks*; see [4]). In this study, the specification of a project is assumed to be given in activity-on-arc (AoA) notation by a directed acyclic graph $D = (\mathbf{N}, \mathbf{A})$, in which $\mathbf{N}$ is the set of nodes representing network 'events', and $\mathbf{A}$ is the set of arcs representing network 'activities'. We assume that there is a single start node 1 and a single terminal node $n$, $n = |\mathbf{N}|$. We define *the project completion time* as the maximum of the realization times of all paths leading from node 1 to node $n$.

* Corresponding author.
*E-mail address:* myao@thu.edu.tw (M.-J. Yao).

0898-1221/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2007.01.036

When the durations of all the activities are constants, project managers may easily calculate the project completion time by the well-known Critical Path Method (CPM) in such a deterministic activity network. However, the duration of an activity is a random variable for most of the cases in the real world, and obviously, the project completion time turns out to be another random variable. Facing such a challenge, project managers should pay serious attention to monitoring the uncertainty involved in stochastic activity networks (SAN). Project managers are highly interested in obtaining the probability density function (*pdf*) of the project's completion time, because it provides full insight into the randomness of the completion of the project, and project managers would have the basis for many subsequent decisions such as bidding, budgeting and scheduling, etc.

This paper focuses on obtaining an approximated *pdf* for the project completion time of a large-size SAN. In this study, we devote ourselves to the application of the techniques of discretization, since our literature review in Section 2 shows that the other approaches are *not* applicable to a *large-size* SAN. Note that this study is concerned with a practical and general case, in which we allow *any type* of *pdf* for the durations of activities in the SAN (which is different from those studies restricting special types of *pdf* for the duration of activities).

The organization of the rest of this paper is summarized as follows. Section 2 surveys the studies on obtaining the *pdf* of the project completion time and reviews the technique of discretization. Importantly, we indicate that there are two opportunities to improve the techniques of discretization when applying them to obtain an approximated *pdf* of the project completion time. Therefore, we propose a new approximation algorithm in Section 3. Section 4 presents our numerical experiments in which we randomly generated 20 sets of 100-node instances for evaluation. Taking the results from a Monte Carlo simulation using 20,000 samples as a benchmark, we compare our proposed approximation algorithm with the PERT model and Dodin's [1] algorithm. Finally, Section 5 gives our concluding remarks.

## 2. Literature review

In this section, the first part surveys the studies on obtaining the *pdf* of the project completion time. The second part reviews the technique of discretization, which is the methodology used in this study. The third part indicates that there are two opportunities to improve the techniques of discretization when applying them to obtain an approximated *pdf* of the project completion time.

### *2.1. The studies on obtaining the pdf of the project completion time*

We note that the project completion time is a random variable that cannot be calculated directly by the summation of all activities on the critical path as in a deterministic activity network, since the duration of all project activities are positive random variables with specified probability distributions. It is well known that the estimation of the cumulative density function (*cdf*) of the completion time of a project is a #-hard problem; see [5] for a further reference. Therefore, it is extremely difficult to calculate the "true" *pdf* of the project completion time, especially, for large-scale SAN.

Researchers have devoted a great deal of effort to propose solution approaches to solve this problem. By referring to Adlakha's [6] and Elmaghraby's [7] classification, one can divide the methodologies in the literature into three categories and review the approaches in each category as follows. (Those papers on the analytical bounding of the *pdf* of the project completion time are skipped since it is not the focus of this study.)

(I) Exact analysis: One must undergo a series of analytical calculations of convolution and max operations between activities to obtain the "true" *pdf* of the project completion time. Two basic issues lead to a possible heavy computational load in exact analysis: first, it requires a multivariate integral for the convolutions, and second, one has to deal with the dependency between the completion time of all paths caused by the shared activities. One may refer to [1,8–11] for the studies in this category. We note that most of their results are not general, since they make very restrictive assumptions. For example, Martin [8] assumed that the *pdf's* of the duration of activity are nominally distributed. Fisher, Saisi and Goldstein's [9] approach applied to only the cases in which the durations of the activities are *independent, and exponentially or general-gamma distributed*. Since the analytical calculations in the approaches using exact analysis are too demanding, it is impossible for project managers to use them for even a medium-size SAN.

(II) Monte Carlo simulations: Monte Carlo simulation is a straightforward methodology that employs a large number of deterministic samples from the simulations of SANs to derive a "close-to-true" *pdf* of the project completion time. One may refer to [12] for an excellent introduction to the Monte Carlo sampling approach. Van Slyke [13] has tried the first straightforward simulation (also known as "crude" Monte Carlo simulation) to derive the *pdf* of the project completion time. In trying to lessen the sampling effort and increase the accuracy of the resulting estimates, the "Conditional Monte Carlo simulation" was developed by Burt and Garman [14], Sigal, Pritsker and Solberg [15] and Dodin [1]. Though the Monte Carlo simulation is the simplest and most intuitive approach, as it can yield the *pdf* of any network no matter how complex it is; but the accuracy requirements can render its computational loading to be too heavy to apply to large-size SAN.

(III) Approximation approaches: According to our discussions above, we know that it is difficult to analytically derive the *pdf* of the project completion time, and a Monte Carlo simulation could be very demanding in its computation (especially for large-size SAN). Therefore, approximation approaches become inevitable and important. Almost all of the approximation approaches use a critical assumption, namely, from the start node to the terminal node, all paths and the activities in SAN are *independent*. Besides, many approximation approaches make further assumptions to simplify their calculations. For example, the PERT (developed by Malcolm et al. [16]) assumes that following the central limit theorem, the duration of each path is normally distributed with its mean and its variance being equal to the sum of the means, and the sum of the variances of the activities on the path. Similar to the PERT model, Sculli [17] assumes that the *pdf* of the activity duration are normal, and the maximum of two normally distributed random variables are also normally distributed (though it is obviously not true). Then, he obtains the first two moments of the maximum using the expressions derived by Clark [18]. Then, one may obtain an approximated *pdf* of the project completion time by *recursively* applying such an approximation scheme to secure the mean and variance of the realization time of any node in the network.

Another category of approximations are the techniques of discretization in which one must replace the continuous *pdf* of activity duration with its discretized version. Dodin [1] proposed an elegantly simple and intuitively appealing procedure, which is the most representative approach among those in this category. (Since we propose an improved approach which is inspired by Dodin's approach, the details of his approach will be presented in Section 2.2.) Later, Dodin and Sirvanci [19] commented that the *pdf* of project completion time tends to converge to the extreme value distribution (rather than the normal distribution) in most cases, under the assumption that all the activities' duration must be *independently* and *identically* distributed.

On the other hand, without assuming the activities to be independent, Mehrotra, Chai and Pillutla [20] divided the SAN into two parts, namely, an independent portion and a dependent portion, and computed the moments of the paths in each portion separately, and finally, combined the two parts together. For large-size SANs, following Dodin and Sirvanci's [19] advocacy, they also approximate the *pdf* of the project completion time using the look-up tables of the *Extreme Value Distribution*.

The focus of this study is to propose a new approach to efficiently obtain the *pdf* of the project completion time. Since it is difficult to apply exact analysis and Monte Carlo simulations to medium-size or large-size SANs, we devote ourselves to approximation approaches in this study. Also, since we are interested in a SAN that allows general types of *pdf* for the duration of activities in this study, we do not consider those approaches that are restricted to using special types of *pdf* for the duration of activities; such as the approaches in [17,19] and [20].

### 2.2. A review on the implementation of techniques of discretization

Here, we use Dodin's [1] approach, which is named as the Dodin Algorithm (DA), to demonstrate the implementation of the techniques of discretization.

When applying the techniques of discretization, the first step is to 'discretize' the *pdf* of the duration of each activity in the activity network by determining the discrete sample points and their corresponding probabilities. In this study, we use the approach of Chebychev sampling points for the discretization of the *pdf* of the duration of each activity. (Agrawal and Elmaghraby [21] introduced three approaches for the implementation of discretization, namely, the approach of Chebychev sampling points, the equal probability approach and the equal distance approach. They commented that the approach of Chebychev sampling points is superior to the other two approaches.)

DA approximates the realization time of any node $j$, denoted as $\gamma_j$, by a series of convolution and max operations over the activities of a SAN as follows. Let $X_{ij}$ be the duration of arc $(i, j)$, *i.e.*, the activity from node $i$ to node $j$,

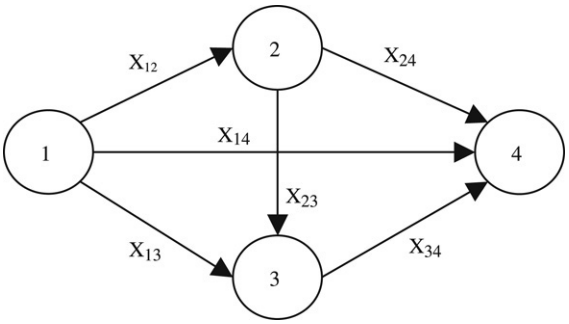


Fig. 1. An AoA network with six activities.

where $i \in B_j$ and $B_j$ denotes the set of nodes immediately preceding node $j$. We define the *pdf* of the realization time at node $j$ via the path from node $i$ to node $j$ by $Y_i^j$, namely,

$$Y_i^j = \gamma_i \oplus X_{ij}, \quad i \in B_j \tag{1}$$

where the symbol $\oplus$ means *convolution* operation. The *cdf* of $Y_i^j$ can be represented by

$$\Pr(Y_i^j \le t) = \int_{-\infty}^{\infty} g_i(t-\tau) f_{ij}(\tau) \mathrm{d}\tau \tag{2}$$

where $g_i(\cdot)$ and $f_{ij}(\cdot)$ denote the *pdf's* of $\gamma_j$ and $X_{ij}$, respectively. Then, DA determines the *pdf* of the realization time of node $j$ using the following max operation:

$$\gamma_j = \max_{i \in B_j} \{Y_i^j\} \tag{3}$$

Finally, DA finishes by obtaining the *pdf* of the project completion time when it reaches the terminal node $n$.

When applying the techniques of discretization, we first need to replace the continuous *pdf* of $X_{ij}$ with its discretized version using the approach of Chebychev sampling points. We represent the discretized *pdf* by an ordered vector as (4):

$$\begin{pmatrix} v_1^{ij} & v_2^{ij} & \cdots & v_m^{ij} \\ p_1^{ij} & p_2^{ij} & \cdots & p_m^{ij} \end{pmatrix} \tag{4}$$

where $v_k^{ij}$ and $p_k^{ij}$ are the value of the $k$th sample point of $X_{ij}$ and its corresponding probability, respectively, and $m$ is the total number of sample points.

Taking the small SAN with six activities in Fig. 1 as an example, one could use DA to approximate the *pdf* of the realization time of $\gamma_4$ (which is also the completion time of the project) using the following six steps: (i) $\gamma_2 = X_{12}$; (ii) $Y_2^3 = \gamma_2 \oplus X_{23}$; (iii) $\gamma_3 = \max\{X_{13}, Y_2^3\}$; (iv) $Y_2^4 = \gamma_2 \oplus X_{24}$ (via the path $\langle 1, 2, 4\rangle$); (v) $Y_3^4 = \gamma_3 \oplus X_{34}$ (via the path $\langle 1, 3, 4\rangle$); and (vi) $\gamma_4 = \max\{X_{14}, Y_2^4, Y_3^4\}$.

In Dodin's [1] paper, he also proposed a procedure, which is called the Discrete Re-sampling Technique (DRT), to save the possible heavy computational loading by re-sampling the *pdf* with a small number of sampling points when the number of sampling points becomes very large (*e.g.*, larger than 100 in our numerical experiments in Section 4). (One should refer to [1] for the details on the DRT.)

### 2.3. Our motivation to improve the techniques of discretization

In this section, we indicate that this study plans to improve the techniques of discretization by: (1) saving the computation loading in max operations; and (2) reducing the bias from assuming independency between paths.

*2.3.1. Save the computation loading in max operations*

This subsection demonstrates that max operations in the techniques of discretization may require significant computational load.

Let $X$ and $Y$ be the realization time of two incidence paths to a node $j$. Then, the realization time of node $j$, which is denoted by a new random variable $Z$, should be obtained by a max operation, *i.e.*, $Z = \max(X, Y)$. Suppose that the number of (discrete) sample points of $X$ and $Y$ are $c_x$ and $c_y$, respectively. Then, we may express the *pdf*'s of $X$ and $Y$ by

$$X = \begin{bmatrix} x_1, x_2, \ldots, x_{c_x} \\ p_1^x, p_2^x, \ldots, p_{c_x}^x \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1, y_2, \ldots, y_{c_y} \\ p_1^y, p_2^y, \ldots, p_{c_y}^y \end{bmatrix} \tag{5}$$

where $x_1 < x_2 < \cdots < x_{c_x}$ and $y_1 < y_2 < \cdots < y_{c_y}$. Also, we may express the (discrete) *pdf* resulting from the max operation in a similar fashion as follows:

$$Z = \begin{bmatrix} z_1, z_2, \ldots, z_{c_z} \\ p_1^z, p_2^z, \ldots, p_{c_z}^z \end{bmatrix} \tag{6}$$

Before analyzing the complexity of the max operation, we present the calculations for a max operation as follows:

(I) For each possible pair of $x_i$ and $y_j$, obtain $z_{ij} = \max(x_i, y_j)$ and its corresponding probability (in which we define $p_{ij}^z = p_i^x p_j^y$ where $i = 1, \ldots, c_x$ and $j = 1, \ldots, c_y$. (There are at most $c_x \cdot c_y$ pairs in such a case.)

(II) Sort the ordered pair $(z_{ij}, p_{ij}^z)$ in ascending order of $z_{ij}$.

(III) Collect those pairs with the same value of $z_{ij}$ and compute the sum of their $p_i^x p_j^y$ values. Then, we obtain the *pdf* of $Z$ expressed in (6).

The first step in a max operation requires $c_x \cdot c_y$ times of comparisons and multiplications. Then, the second step needs $(c_x \cdot c_y) \ln(c_x \cdot c_y)$ times of comparisons to obtain the sorted sequence in the worst case. Finally, the third step could be finished within $c_x \cdot c_y$ times of making comparisons using the sorted sequence from the second step.

Note that the number of sampling points (*e.g.*, the values of $c_x$ and $c_y$) usually grow quickly during the process of the approximation. For most of the practical cases, the sampling points are of *real* values (rather than integer values) when applying the techniques of discretization. Particularly, when applying a convolution operation to two random variables $X_1$ and $X_2$, the number of sampling points for the resulting *pdf* of $X_1 \oplus X_2$ is often close to $c_{x_1} \cdot c_{x_2}$, since the resulting values of the sampling points are different from each other. Therefore, after several convolution operations, the number of sampling points for any one of the paths in a SAN could be large. Obviously, since both $c_x$ and $c_y$ may be of large values, the computations in a max operation could become time-consuming. Therefore, we need to derive a new heuristic to approximate the computations of max operation so as to improve the efficiency of the techniques of discretization.

*2.3.2. The bias from assuming independence between paths*

In practical cases, the sub-paths in a SAN are *not* independent, since they usually share activities with each other. However, most of the approaches for obtaining the *pdf* of the project completion time in the literature did not take into account the dependency between paths (or sub-paths). In this subsection, we will show that there may exist *significant* bias in the approximated *pdf* of the project completion time if one does not consider path dependency in the applications of techniques of discretization.

First, we show that *the number of the shared activities* could be used to measure *the dependence between two paths* using a SAN with seven activities in Fig. 2. Also, the path dependency may generate a significant bias in the estimate of *mean* of the project completion time. We assume that the duration of each activity in Fig. 2 is exponentially distributed with their means being $\mu(X_{12}) = 1$, $\mu(X_{23}) = 3$, $\mu(X_{34}) = 5$, $\mu(X_{45}) = 2$, $\mu(X_{46}) = 2$, $\mu(X_{57}) = 3$, and $\mu(X_{67}) = 3$, respectively.

Similar to other techniques of discretization, DA considers the paths $\langle 1, 2, 3, 4, 5, 7\rangle$ and $\langle 1', 2', 3', 4', 6, 7\rangle$ as two *independent* paths as shown in Fig. 3. For those two "independent" paths, we know that $X_{12} = X_{1'2'}$, $X_{23} = X_{2'3'}$, $X_{34} = X_{3'4'}$, $\gamma_6 = \gamma_{4'} \oplus X_{4'6}$, $Y_6^7 = \gamma_6 \oplus X_{67}$, and $\gamma_7 = \max\{Y_5^7, Y_6^7\}$.

In the following experiment, we will show that before applying the max operation to two paths at the last node, one could obtain excellent estimates for the mean of the completion time of the paths. (In such a case, for these two

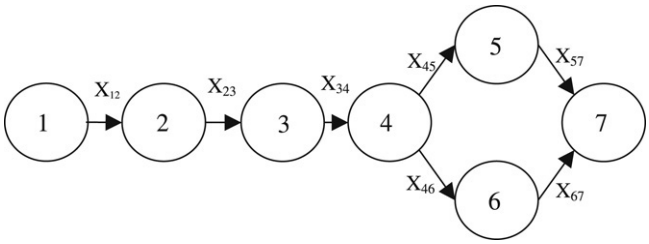


Fig. 2. An AoA network with seven activities.

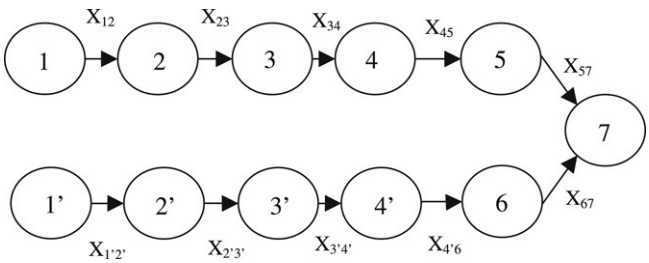


Fig. 3. An equivalent network of the AN in Fig. 2.

Table 1
The first moments of the *pdf*'s with different ending nodes in Fig. 2

| Ending node | MCS | DA | | The proposed algorithm | |
|---|---|---|---|---|---|
| | Mean | Mean | Error (%) | Mean | Error (%) |
| 3 | 3.98 | 4.00 | 0.50 | 4.00 | 0.50 |
| 4 | 9.07 | 9.00 | 0.77 | 9.00 | 0.77 |
| 5 | 11.05 | 11.05 | 0.45 | 11.05 | 0.45 |
| 7 | 15.89 | **17.69** | **11.33** | **15.57** | **2.01** |

paths, their *pdf*'s could be easily obtained by a series of convolution operations.) By assuming that two paths are *independent*, we observe *significant* errors existing in the estimate of the mean (of the resulting *pdf*) after applying the max operation to two paths.

For the example in Fig. 2, the third and the fourth columns (*i.e.*, under the approach of DA) of Table 1 summarize the measures of the mean of the *pdf*'s and its error or the paths with different ending nodes. We note that the "ending node" in Table 1 indicates the corresponding path; for example, when the "ending nodes" are 3 and 4, the corresponding paths are $\langle 1, 2, 3\rangle$ and $\langle 1, 2, 3, 4\rangle$, respectively; and so on. One may observe that we obtain excellent estimates for the mean for the cases with the ending node being 3, 4, 5 or 6. However, significant errors exist when both paths end at node 7.

Closely examining Fig. 3, we can get a better idea about the source of the error. Though it is true that $X_{12} = X_{1'2'}$, $X_{23} = X_{2'3'}$, $X_{34} = X_{3'4'}$, DA (like other techniques of discretization) treats $\langle 1, 2, 3, 4\rangle$ and $\langle 1', 2', 3', 4'\rangle$ as two *different* sub-paths. When applying a max operation on the two "independent" paths $\langle 1, 2, 3, 4, 5, 7\rangle$ and $\langle 1', 2', 3', 4', 6, 7\rangle$ at node 7, we distort the computations of the discrete *pdf* by "double counting" the part from $\langle 1', 2', 3', 4'\rangle$. Consequently, the mean of the resultant *pdf* of the realization time of node 7 would become significantly larger than that from MCS. We name such an error the Shared Activity Bias (SAB), since it results from the shared activities between paths in SAN according to our discussion above. Therefore, before applying a max operation to two paths in a SAN, we are motivated to check if there exist shared activities between them so as to eliminate the SAB.

## 3. The proposed approximation algorithm

This section proposes two new components to revise DA for approximating the *pdf* of the project completion time in SAN. These two new components will improve DA by: (1) saving the computation loading in max operations by

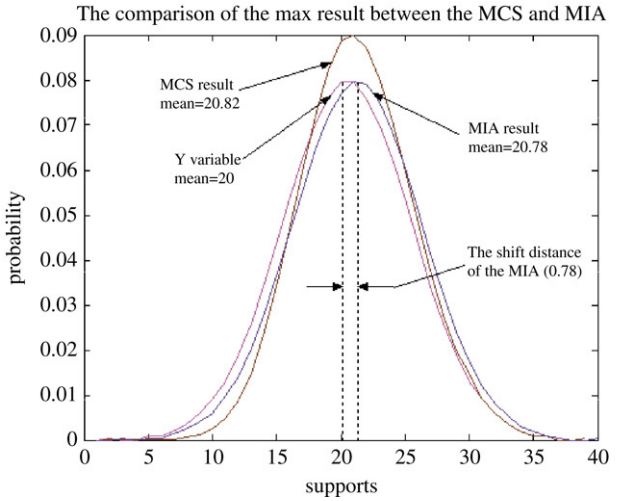


Fig. 4. The results of a max operation from the MCS and the approximation algorithm.

an approximation procedure; and (2) reducing the SAB using a simple heuristic rule. We will present the analysis and implementation of those two new components in the following two subsections.

### *3.1. An approximation procedure to replace the max operation*

In this section, we propose a new approximation procedure to replace max operation in the techniques of discretization.

First, we explain the rationale for our approximation procedure by using a figure. Suppose that we have two *independent* random variables $X$ and $Y$ being normally distributed with their *pdf's* being $N(14, 25)$ and $N(20, 25)$, respectively. Let $Z = \max(X, Y)$, and we approximate the *pdf* of $Z$ by 20,000 times of MCS, which is shown in Fig. 4. It is interesting that the *pdf* of $Z$ is very similar to that of the *pdf* of $Y$, except the kurtosis (or peakedness) of the former is larger.

Note that the mean of the *pdf* of $Z$ is no less than that of $Y$. (One may refer to theory of order statistics and other studies, *e.g.*, Mehrotra, Chai and Pillutla [20], Klingel Jr. [22] and Pontrandolfo [23] for further references on such an assertion.) That is, $E(Z) = E(Y) + RSV$ where $RSV$ is a Right-Shifting-Value ($RSV$) from $E(Y)$ to $E(Z)$ and $RSV \geq 0$. Such a phenomenon inspires an appealing idea: *if we could estimate the value of the RSV using simple calculations, then we could approximate the pdf of Z by shifting the pdf of Y to the right by a magnitude of RSV*. In such a case, we may save significant computational loading by replacing the calculations for a max operation with the simple calculations to obtain the value of the $RSV$. Following this clue, we devised an effective procedure for obtaining the value of $RSV$, which is discussed next.

In the follow discussion, we use the notation for the random variables $X$, $Y$ and $Z$ the same way as those defined in Section 2.3.1. To facilitate our presentation, we use a small example with $X = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{bmatrix}$ and $Y = \begin{bmatrix} 4 & 6 & 8 & 10 & 12 & 14 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$. The calculations of the max operation to obtain the *pdf* of $Z$ are summarized in Table 2. Using the max operation, one should calculate the expected value of $Z$ by the following expression:

$$E(Z) = \sum_{i=1}^{c_x} \sum_{j=1}^{c_y} \max(x_i, y_j) p_i^x p_j^y = \sum_{i=1}^{c_x} \sum_{j=1}^{c_y} z_{ij} p_i^x p_j^y \tag{7}$$

We define two variables that are important in the following discussions: $x_s \triangleq \min_{i=1,\ldots,c_x}\{x_i \geq y_1\}$ and $y_k \triangleq \min_{j=1,\ldots,c_y}\{y_j \geq x_{c_x}\}$. By referring to Table 2, it holds that

$$E(Z) = (\text{gray area}) + (\text{white area}) \tag{8}$$

where (gray area) results from the condition $\max(x_i, y_j) = x_i$ and (white area) from $\max(x_i, y_j) = y_j$.

Table 2
The computation of the max operation $Z = \max(X, Y)$

| Y | Pr($Y$) | X | | | | |
|---|---|---|---|---|---|---|
| | | $x_1 = 1$ | $x_2 = 3$ | $x_3 = 5$ | $x_4 = 7$ | $x_5 = 9$ |
| | | Pr($X$) | | | | |
| | | $p_1^x$ | $p_2^x$ | $p_3^x$ | $p_4^x$ | $p_{c_x}^x$ |
| $y_1 = 4$ | $p_1^y$ | $\max\{y_1, x_1\} = y_1 = 4$ $p_1^y p_1^x$ | $\max\{y_1, x_2\} = y_1 = 4$ $p_1^y p_1^x$ | $\max\{y_1, x_3\} = x_3 = 5$ $p_1^y p_3^x$ | $\max\{y_1, x_4\} = x_4 = 7$ $p_1^y p_4^x$ | $\max\{y_1, x_{c_x}\} = x_5 = 9$ $p_1^y p_{c_x}^x$ |
| $y_2 = 6$ | $p_2^y$ | $\max\{y_2, x_1\} = y_2 = 6$ $p_2^y p_1^x$ | $\max\{y_2, x_2\} = y_2 = 6$ $p_2^y p_2^x$ | $\max\{y_2, x_3\} = y_2 = 6$ $p_2^y p_3^x$ | $\max\{y_2, x_4\} = x_4 = 7$ $p_2^y p_4^x$ | $\max\{y_2, x_{c_x}\} = x_5 = 9$ $p_2^y p_{c_x}^x$ |
| $y_3 = 8$ | $p_3^y$ | $\max\{y_3, x_1\} = y_3 = 8$ $p_3^y p_1^x$ | $\max\{y_3, x_2\} = y_3 = 8$ $p_3^y p_2^x$ | $\max\{y_3, x_3\} = y_3 = 8$ $p_3^y p_3^x$ | $\max\{y_3, x_4\} = y_3 = 8$ $p_3^y p_4^x$ | $\max\{y_3, x_{c_x}\} = x_5 = 9$ $p_3^y p_{c_x}^x$ |
| $y_4 = 10$ | $p_4^y$ | $\max\{y_4, x_1\} = y_4 = 10$ $p_4^y p_1^x$ | $\max\{y_4, x_2\} = y_4 = 10$ $p_4^y p_2^x$ | $\max\{y_4, x_3\} = y_4 = 10$ $p_4^y p_3^x$ | $\max\{y_4, x_4\} = y_4 = 10$ $p_4^y p_4^x$ | $\max\{y_4, x_{c_x}\} = y_4 = 10$ $p_4^y p_{c_x}^x$ |
| $y_5 = 12$ | $p_5^y$ | $\max\{y_5, x_1\} = y_5 = 12$ $p_5^y p_1^x$ | $\max\{y_5, x_2\} = y_5 = 12$ $p_5^y p_2^x$ | $\max\{y_5, x_3\} = y_5 = 12$ $p_5^y p_3^x$ | $\max\{y_5, x_4\} = y_5 = 12$ $p_5^y p_4^x$ | $\max\{y_5, x_{c_x}\} = y_5 = 12$ $p_5^y p_{c_x}^x$ |
| $y_6 = 14$ | $p_{c_y}^y$ | $\max\{y_{c_y}, x_1\} = y_6 = 14$ $p_{c_y}^y p_1^x$ | $\max\{y_{c_y}, x_2\} = y_6 = 14$ $p_{c_y}^y p_2^x$ | $\max\{y_{c_y}, x_3\} = y_6 = 14$ $p_{c_y}^y p_3^x$ | $\max\{y_{c_y}, x_4\} = y_6 = 14$ $p_{c_y}^y p_4^x$ | $\max\{y_{c_y}, x_{c_x}\} = y_6 = 14$ $p_{c_y}^y p_{c_x}^x$ |

Therefore, one may express (gray area) and (white area) by the following equations:

$$\text{(gray area)} = \sum_{i=(s+j-1)}^{c_x} \sum_{j=1}^{k} \max(x_i, y_j) p_i^x p_j^y = \sum_{i=(s+j-1)}^{c_x} \sum_{j=1}^{k} x_i p_i^x p_j^y \tag{9}$$

$$\begin{aligned} \text{(white area)} &= \sum_{i=1}^{(s+j-2)} \sum_{j=1}^{(k-1)} \max(x_i, y_j) p_i^x p_j^y + \sum_{i=1}^{c_x} \sum_{j=k}^{c_y} \max(x_i, y_j) p_i^x p_j^y \\ &= \sum_{j=1}^{(k-1)} \sum_{i=1}^{(s+j-2)} y_j p_i^x p_j^y + \sum_{j=k}^{m} \sum_{i=1}^{n} y_j p_i^x p_j^y \end{aligned} \tag{10}$$

Recall that we estimated the value of *RSV* using the following equation:

$$RSV = E(Z) - E(Y) \tag{11}$$

Also, since the (white area) is identical for both $E(Z)$ and $E(Y)$, we may obtain the approximated *RSV* from calculating the difference of $E(Z)$ and $E(Y)$ only (in the gray area) by (12).

$$\begin{aligned} RSV &= \sum_{j=1}^{k} RSV_j(y_j) \\ &= \sum_{j=1}^{k} \sum_{i=s+j-1}^{c_x} (x_i - y_j) p_j^y p_i^x \end{aligned} \tag{12}$$

where $RSV_j(y_j) = \sum_{i=s+j-1}^{c_x} (x_i - y_j) p_j^y p_i^x$ is the part of *RSV* corresponding to $y_j$, $j = 1, \ldots, k$.

For the small example in Table 2, one can easily obtain the approximated value of *RSV* to be 0.47 after summing the values of $(x_i - y_j) p_j^y p_i^x$ of 6 cells in the (gray area). Then, following the proposed approximation, one may obtain

an approximated *pdf* of $Z$ by shifting the *pdf* of $Y$ to the right by the approximated value of the *RSV*. Therefore, we obtain the approximated *pdf* of $Z$ by $Z = \begin{bmatrix} 4.47 & 6.47 & 8.47 & 10.47 & 12.47 & 14.47 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$ with the estimate of the mean being 9.4667.

On the other hand, using max operation to obtain the *pdf* of $Z$, one has $Z = \begin{bmatrix} 4 & 5 & 6 & 7 & 8 & 9 & 10 & 12 & 14 \\ 1/15 & 1/30 & 1/10 & 1/15 & 2/15 & 1/10 & 1/6 & 1/6 & 1/6 \end{bmatrix}$ with the estimate of the mean being 9.4667. In this example, the estimate of the mean from the proposed approximation procedure matches exactly with that from max operation. Also, since we compute only 6 cells (in the gray area) rather than all of the 30 cells, we save around 80% of the computational loading in this example.

Compared to the max operation, the proposed approximation procedure may save significant computational loading due to the following two facts:

(I) To work out a max operation, it requires calculating all of the $c_x \cdot c_y$ pairs of $x_i$ and $y_j$. (Please refer to Section 2.3.1 for the details.) Using the proposed approximation procedure, one needs only to compute the (gray) area which is usually less than 20% of the computations for max operation.

(II) When applying a max operation to obtain the *pdf* of $Z = \max(X, Y)$, the number of sampling points of the *pdf* of $Z$ is often significantly larger than that of $Y$. Therefore, if one keeps using max operations in the techniques of discretization, the number of sampling points for the *pdf*'s of the realization time of nodes in SAN usually grows very fast, and it takes a longer run time to approximate the *pdf* of project completion time. On the other hand, by directly using the *pdf* of $Y$ (and shifting it to the right by the *RSV*) to approximate that of $Z$, we observe that the number of sampling points grows in a milder fashion when applying the techniques of discretization.

Due to these two facts, the proposed approximation procedure helps to save significant run time in approximating the *pdf* of project completion time without losing considerable precision at the same time for large-sized instances. (We will present our numerical experiments in Section 4 later.)

After introducing the proposed approximation procedure to save the significant computational loading from the max operation, we proceed to deal with the shared activity bias from assuming independency between paths.

### *3.2. A label-correcting tracing algorithm for eliminating shared activity bias*

Recall that the Shared Activity Bias (SAB) results from the shared activities between paths in SAN according to the discussion in Section 2.3.2. Also, before applying a max operation to two paths, one should check if there exist shared activities between them so as to eliminate the SAB.

Therefore, when applying a max operation, one should be able to take care of the following issues:

(I) One should be able to detect if a particular activity is (or a subset of activities are) shared by two paths (or several paths) in the activity network.

(II) One needs an alternative approach to 'correct' the computations in a max operation so as to eliminate SAB.

In this study, we have revised a Label-Correcting Tracing Algorithm (LCTA) to address the first issue. The authors proposed the LCTA for the implementation of the Dodin [1] algorithm in their previous study, namely, Yao, Chu and Tseng [24]. The LCTA is actually a variant of the label-correcting (LC) approach for determining the shortest path (SP) in a deterministic network. In the LC approach for SP, each node $i$ is assigned a label (usually a tentative distance to the terminal node $n$) and will be put into a queue. Then, the LC approach scans through the nodes in the queue and updates (or corrects) the labels if necessary. After a node is processed, the node will be removed from the queue, and then some nodes may be inserted or re-positioned in the queue. The LC approach repeats the process until the queue is empty, at which point all labels are correct (that is, the SP is found). There are several LC approaches in the literature, including the first-in-first-out algorithm of Bellman [25], the d-queue method of Pape [26], the two-queue method of Gallo and Pallottino [27], and the threshold methods of Glover, et al. [28]. Recently, Bertsekas [29] proposed a small-label-first principle to modify the approaches of Gallo and Pallottino [27] and Glover, et al. [28], making them even faster. The proposed LCTA is different from these LC approaches in two aspects. First, obviously, the LCTA pursues the *longest* path rather than the shortest path in the network concerned; and second, the LCTA uses a data structure of *stack* (rather than data structures of *queue*) for the implementation of the label-correcting process.

When applying the LCTA, we employed two tracing procedures, *viz.*, a Downward_Tracing procedure and an Upward_Tracing procedure, to keep the tracing sequence (using a stack structure) and to implement the corresponding

calculations (*i.e.*, convolution and/or max) and to record the *pdf* of the realization time of each node. In the LCTA, we designated a *Shared_flag*, which is a Boolean variable, in both tracing procedures for identifying if any redundant paths exist in the SAN. During the implementation of the LCTA, the Upward_Tracing procedure first checks the value of the *Shared_flag* for the current treated node; if the current node has been visited previously and the *pdf* of its realization time has been obtained (which means the *pdf* of the duration from the starting node to the current node has been obtained), the *Shared_flag* will be set as 1 by the Downward_Tracing procedure. Using the values of *Shared_flag* and those two tracing procedures, one can find all of the activities shared by the sub-paths in the SAN.

Now, the second issue can be dealt with by applying our heuristic to 'correct' the computations in a max operation so as to eliminate SAB. Our heuristic is as follows: when a redundant path is found, we replace the *pdf* of this redundant path with *the value of its mean* (which is a fixed constant) in the Upward_Tracing procedure. Our experiments in Section 4 show that using such a simple heuristic (of replacing with its mean) could effectively eliminate the SAB.

One may refer to Appendix for the application of the LCTA to the example in Fig. 2. We summarize the results of our new approach in the last two columns (*i.e.*, under the name of "The proposed algorithm") of Table 1. In the part with the "Ending node" being 7, the mean from the DA is 17.59, and the error is 11.33% compared to the MCS. On the other hand, the mean of the proposed approach in this study is 15.57, which is *only 2.01%* compared to the MCS! Based on our illustration in this example, we demonstrate that our heuristic successfully takes care of the error from assuming independency between sub-paths by eliminating Shared Activity Bias (SAB).

## 4. Numerical experiments

This section verifies the efficiency of the proposed approximation algorithm using random experiments. Since we are interested in a SAN that allows general types of *pdf* for the duration of activities in this study, we do not consider those approaches that restrict their use to special types of *pdf* for the duration of activities; such as the approaches in Sculli [17], Dodin and Sirvanci [19] and Mehrotra, Chai and Pillutla [20]. Therefore, in our numerical experiments, we evaluate the effectiveness of the three (general-type) approximation approaches, *viz.*, the proposed approximation algorithm, Dodin Algorithm (DA,) and the PERT model from three aspects: the precision of the *mean* and the *goodness-of-fit* of the approximated *pdf* and their *run time*.

Before presenting the numerical results, we introduce the settings of our random experiments. Following the rules given in Kolisch and Sprecher [30], we randomly generated our 20 instances of 100-node activity networks by setting the level of network complexity factor as 2.0. In our experiments, the number of activities in an instance is in the range of [130, 250]. (By fixing the number of nodes at 100, the total number of activities in an instance is actually a random number.)

One may refer to Fig. 5 for a 100-node instance with 135 activities in our numerical experiments. (We note that our experimental instances are of the largest number of activities among all the studies in the literature.) Also, we randomly designated the duration of each activity to be *normally*, *exponentially* or *uniformly distributed* with its mean being randomly determined in the range of [1, 10]. For each node in the activity network, the number of the preceding (or the successor) nodes is randomly determined between 1 and 3. We have conducted our experiments using a PC with a Pentum-IV 2.0 GHZ CPU and 256 MB RAM. Our program was coded by the Matlab Ver. 6.5 package.

In our experiments, we employ the *pdf* resultant from a MCS using 20,000 samples as a benchmark for comparison, since it is extremely difficult to obtain the "true" *pdf* of the project completion time for these 20 sets of 100-node instances. (Based on our experimental results from these 20 instances, the *standard error* of the mean from MCS is given by 0.36% on average. Therefore, the *pdf* resultant from a MCS using 20,000 samples should be precise enough to serve as a benchmark for comparison. Also, please refer to [1,14] and [15] for the supports on using MCS as a benchmark for comparison.)

To evaluate the performance of the three approximation approaches, we will compare them in the following three aspects: the run time of the algorithm, the error in the mean of the approximated *pdf's* (comparing to the *pdf* resultant from MCS using 20,000 samples) and the number of instances that pass K–S tests among the 20 instances. Let $\bar{X}_{Y\&C}$ be the mean of the proposed approximation algorithm (with $Y\&C$ indicating the initials of the authors' last names Yao and Chu). We define "the error in the mean" for the three approximation approaches by $|DA - MCS|/MCS$, $|PERT - MCS|/MCS$ and $|MIA - MCS|/MCS$, respectively, where $\bar{X}_{DA}$, $\bar{X}_{PERT}$ and $\bar{X}_{MCS}$ are the means obtained by

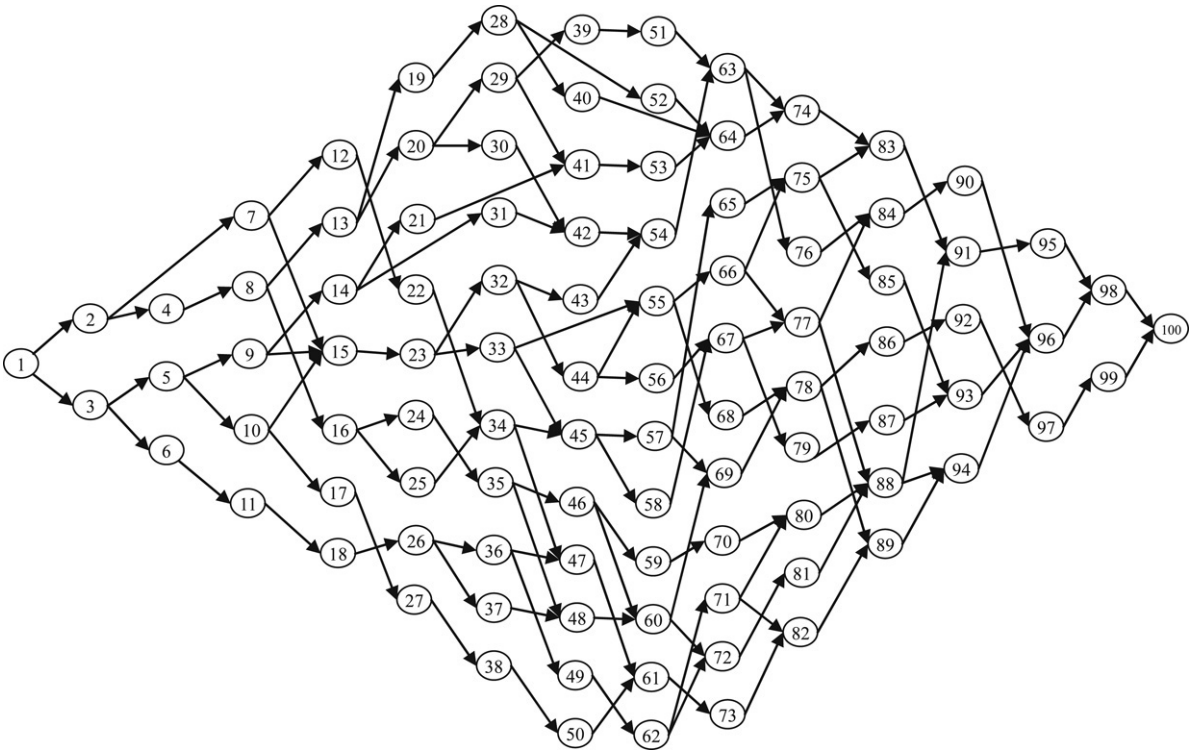

Fig. 5. A 100-node instance with 135 activities in our numerical experiments.

Table 3
A summary of the comparison in the run time (in seconds), the error in mean and the K–S tests of our algorithm ($Y\&C$), DA and PERT of 20 instances with 100 nodes

| Comparison criteria | MCS | $Y\&C$ w/o DRT | $Y\&C$ w/ DRT | DA w/o DRT | DA w/DRT | PERT |
|---|---|---|---|---|---|---|
| Avg. run time (in seconds) | 2103.70 | 1.71 | 0.76 | 137.47 | 2.76 | 0.12 |
| Avg. error in the mean (%) | – | 2.35 | 2.42 | 26.55 | 25.46 | 23.99 |
| K–S test with $\alpha = 0.01$ | – | 20/20 | 20/20 | 0/20 | 0/20 | 0/20 |

the DA, the PERT and the MCS, respectively. To facilitate our using look-up tables, we first apply the Discrete Re-sampling Technique (DRT) to re-arrange the approximated *pdf's* using the designated number of sample points before employing K–S test for comparing their goodness-of-fit. Here, we conduct our K–S tests using 30 sample points with a critical value of $\alpha = 0.01$. Table 3 summarizes our experimental results for the three approximation approaches.

From Table 3, MCS is obviously very time consuming and it takes over 2,100 s on average to obtain the *pdf* of the project completion time for our 20 instances. The DA needs 137.47 s and 2.76 s on average for the versions without and with the use of the DRT, respectively. On the other hand, without using the DRT, the average run time of the proposed approximation algorithm is 1.71 s. After applying the DRT to the proposed approximation algorithm, its average run time is trimmed down to only 0.76 s. Impressively, for both the DA and the proposed approximation algorithm, the application of the DRT dramatically improves its run time. Therefore, we conclude that the DRT plays a crucial role to improve the effectiveness of the techniques of discretization based on our experimental results.

Also, we would like to indicate an interesting observation, namely, the average run time of the proposed approximation algorithm (*i.e.*, $Y\&C$ in Table 3) is *5 times faster* than that of the DA when both of them apply the DRT. Recall that the proposed approximation algorithm replaces the max operation with an approximation procedure (presented in Section 3.1) and directly sets the *pdf* of a redundant path to be *the value of its mean* (which is a constant with a probability being equal to 1) when detecting a set of shared activities. These two components assist us to reduce

the computational load by preventing the rapid growth in the number of sampling points in the calculations used in techniques of discretization. It also explains the reason why the proposed approximation algorithm is much more effective than the DA in the running time aspect.

One may be curious about "How good the *pdf* obtained by the PERT is" since its average run time is only 0.12 s, which is extremely short. Table 3 shows that the average error in the mean of the *pdf* obtained by the proposed approximation algorithm is as low as 2.35% and 2.42% for the versions without and with applying the DRT, respectively. Therefore, we assert that the proposed approximation algorithm is able to obtain an excellent estimate of the *expected project completion time* (with or without using the DRT). Though the running time of the PERT approach is impressively fast, it obtains a very poor estimate of the mean with an average error of 23.99% which would surely make those interested project managers very disappointed!

The last row of Table 3 presents five sets of K–S tests. Apparently, the PERT model and the DA produce inferior approximated *pdf's*, since no one out of these 60 K–S tests accept the approximated *pdf's* from both approaches being the same as that resultant from a MCS using 20,000 samples. On the other hand, the approximated *pdf's* obtained by the proposed approximation algorithm pass all of the 40 K–S tests.

Based on our experimental results, we assert that the proposed approximation algorithm significantly outperforms the DA and the PERT in both their running time and the precision aspects. Also, we advocate the application of the DRT when implementing the technique of discretization.

## 5. Conclusion

When facing stochastic activity networks (SANs), the project managers are interested to secure the *pdf* of the project completion time so as to have complete insight into the randomness of the realization of the project. For large-sized SANs, the project managers must turn to the techniques of discretization, since the other approaches in the literature become too demanding in terms of computational loading to obtain the *pdf* of the project completion time. Recall that in Section 3, we indicated two opportunities to improve the techniques of discretization, namely, (1) to saving the computation loading in max operations by an approximation procedure; and (2) to reduce the Shared Activity Bias (SAB) using a simple heuristic rule. Therefore, we are motivated to propose our approximation algorithm in this study. To evaluate the performance of the proposed algorithm, we randomly generated 20 sets of 100-node instances. Using the results from our Monte Carlo simulation (MCS) using 20,000 samples as our benchmarks, we compared the proposed algorithm with the PERT model and Dodin's [1] algorithm (DA). Our experimental results concluded that the proposed algorithm significantly outperforms the DA and the PERT in both their running time and the precision aspects. Also, we learned that the discrete re-sampling technique (DRT) is very important to save running time (without significant loss in its precision) when applying the technique of discretization.

Another important contribution of this study is to provide an effective tool, namely, the proposed approximation algorithm, for obtaining an accurate estimate for the mean of the *pdf* of the project completion time for the SAN that allows *general types* of *pdf* for the duration of the activities. For many optimization problems in SAN, *e.g.*, the time-cost trade-off problems and the resource allocation problems, it is a critical component to solve these optimization problems in order to be able to obtain an accurate estimate for the mean of the *pdf* of the project completion time. (Recall that most of the studies derived the estimates of the mean based on the strong assumptions of using some specific types of distributions according to our literature review in Section 2.) By referring to our numerical experiments in Section 4, the proposed approximation algorithm obtains excellent estimates for the mean of the *pdf* of the project completion time with an average error of only 2.42%, and an average completion time of only 0.76 s for those 100-node instances. Therefore, the proposed algorithm would serve as an effective tool for solving these optimization problems.

## Acknowledgements

This research is supported by the National Science Council, Taiwan, Republic of China (grant NSC-95-2221-E-029-026-MY2). The authors would like to thank the referee and Prof. E.S. Lee for their valuable comments, which helped in improving this paper.

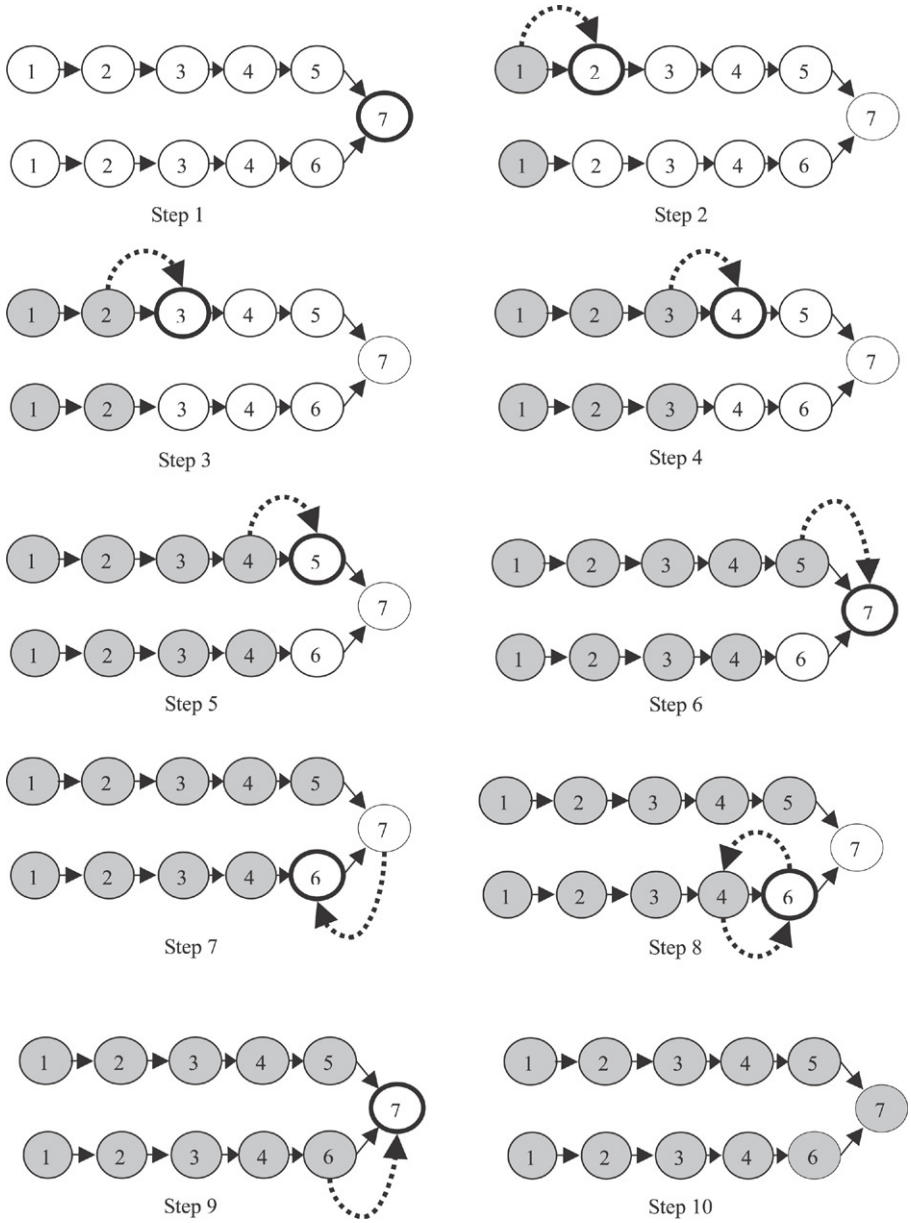


Fig. 6. The steps of the MIA when applying it to the AN in Fig. 2.

## Appendix. The application of the LCTA to the example in Fig. 2

Here, we use the example in Fig. 2 to demonstrate how to apply the LCTA to eliminate SAB. To facilitate our presentation, Fig. 6 summarizes the step-by-step details of the LCTA in this example.

The LCTA starts from the terminal node, *i.e.*, node 7, using the Downward_Tracing procedure, and it traces all the way to node 1, which is the beginning of the path $\langle 1, 2, 3, 4, 5, 7\rangle$ in the first step. (The LCTA employs a *stack* data structure to maintain the nodes passed in the Downward_Tracing procedure.) In Step 2, we first set $\gamma_1 = 0$ and label it gray (which indicates that we secured its realization time), since there is no proceeding node at node 1. Then, since node 2 is the one on the top of the stack, we move to node 2 by the Upward_Tracing procedure. Next, in Step 3, we first obtain the realization time of node 2 via the activity $X_{12}$ by $Y_1^2 = \gamma_1 \oplus X_{12} = X_{12}$. Also, since there is no branching node other than node 1, we have the realization time of node 2 by $\gamma_2 = Y_1^2$. Steps 4 to 6 follow the same way to obtain

$\gamma_3$, $\gamma_4$ and $\gamma_5$. (Note that we label the nodes 1, 2, 3, 4 and 5 gray before Step 7 since we have obtained the realization time so far.) At the end of Step 6, we use the Upward_Tracing procedure to reach node 7 and obtain $Y_5^7 = \gamma_5 \oplus X_{57}$. At Step 7, we find another incidence path entering into node 7. Therefore, we employ the Downward_Tracing procedure to reach node 6. At Step 8, when trying to use Downward_Tracing procedure at node 6 to reach node 4, we detect that node 4 has been labeled gray, which indicates that the realization time of node 4 has been obtained. This means that *a subset of activities, which is the redundant sub-path* ⟨1', 2', 3', 4'⟩, *are shared by two paths*! Then, we set the value of *Shared_flag* as 1. Highlighting by these steps, we have shown that the LCTA is able to detect *redundant sub-paths* in the SAN.

Now, we are ready to apply our heuristic to eliminate SAB. Since the sub-paths ⟨1, 2, 3, 4⟩ and ⟨1', 2', 3', 4'⟩ are actually the same, we compute the *pdf* of the sub-path ⟨1', 2', 3', 4', 6⟩ by $Y_4^6 = \mu(\gamma_4) \oplus X_{46}$. Note that in the convolution operation, we replace the *pdf* of $\gamma_{4'}$ with $\gamma_{4'} = \begin{bmatrix} \mu(\gamma_4) \\ 1 \end{bmatrix}$ to avoid the error from double-counting the discrete *pdf* of ⟨1, 2, 3, 4⟩.

Before comparing the results from our approach with those from MCS, we finish the rest of the computation in this example. Since there is no branching node other than node 4, we have the realization time of node 6 by $\gamma_6 = Y_4^6$. At Step 9, we use the Upward_Tracing procedure to reach node 7 from node 6, and compute $Y_6^7 = \gamma_6 \oplus X_{67}$. Finally, we observe that no other branching node from node 7 needs to compute its realization time at Step 10. We may obtain the realization time of node 7, which is the project completion time in this example, by a max operation, *i.e.*, $\gamma_7 = \max(Y_5^7, Y_6^7)$.

## References


[1] B.M. Dodin, Approximating the distribution function in stochastic networks, Comput. Oper. Res. 12 (3) (1985) 251–264.
[2] J. Coppendale, Manage risk in product and process development and avoid unpleasant surprises, Eng. Manage. J. 5 (1) (1995) 35–38.
[3] P. Chatzoglou, L. Macaulay, A review of existing models for project planning and estimation and the need for a new approach, Int. J. Proj. Manage. 14 (1996) 173–183.
[4] S.E. Elmaghraby, Activity Networks: Project Planning and Control by Network Models, Wiley, New York, 1977.
[5] J.N. Hagstorm, Computational complexity of PERT problems, Networks 18 (1988) 139–147.
[6] V.G. Adlakha, A classified bibliography of research on stochastic PERT networks: 1966–1987, INFOR 27 (3) (1989) 272–296.
[7] S.E. Elmaghraby, The estimation of some network parameters in the PERT model of activity networks: Review and critique, in: R. Slowinski, J. Weglarz (Eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp. 371–432.
[8] J.J. Martin, Distribution of the time through a directed acyclic networks, Oper. Res. 13 (1) (1965) 46–66.
[9] D.L. Fisher, D. Saisi, W.M. Goldstein, Stochastic PERT networks: OP diagrams critical paths and the project completion time, Comput. Oper. Res. 12 (5) (1985) 471–482.
[10] V.G. Kulkarni, V.G. Adlakha, Markov and Markov-Regenerative PERT networks, Oper. Res. 34 (5) (1986) 769–781.
[11] J.N. Hagstrom, Computing the probability distribution of project duration in a PERT network, Networks 20 (1990) 231–244.
[12] J.R. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer Verlag, New York, 1997.
[13] R.M. van Slyke, Monte Carlo methods and the PERT problem, Oper. Res. 11 (1963) 839–861.
[14] J.M. Burt, M.B. Garman, Conditional Monte Carlo: A simulation technique for stochastic network analysis, Manage. Sci. 18 (3) (1971).
[15] C.E. Sigal, A.B. Pritsker, J.J. Solberg, The use of cutsets in Monte Carlo analysis of stochastic networks, Math. Comput. Simuation 21 (1979) 376–384.
[16] D.G. Malcolm, J.H. Roseboom, C.E. Clark, W. Fazar, Application of a technique for research and development program evaluation, Oper. Res. 7 (1959) 646–669.
[17] D. Sculli, The completion time of PERT networks, J. Oper. Res. Soc. 25 (1) (1983) 155–158.
[18] C.E. Clark, The greatest of a finite set of random variables, Oper. Res. 9 (1961) 146–162.
[19] B.M. Dodin, M. Sirvanci, Stochastic networks and the extreme value distribution, Comput. Oper. Res. 17 (4) (1990) 207–217.
[20] K. Mehrotra, J. Chai, S. Pillutla, A study of approximating the moments of the job completion time in PERT networks, J. Oper. Manage. 14 (1996) 277–289.
[21] M.K. Agrawal, S.E. Elmaghraby, On computing the distribution function of the sum of independent random variables, Comput. Oper. Res. 28 (2001) 473–483.
[22] A.R. Klingel Jr., Bias in PERT project completion time calculations for a real network, Manage. Sci. 13 (4) (1966) 476–489.
[23] P. Pontrandolfo, Project duration in stochastic networks by the PERT-path technique, Int. J. Proj. Manage. 18 (2000) 215–222.
[24] M. Yao, W. Chu, T. Tseng, A label-correcting tracing algorithm for the approximation of the probability distribution function of the project completion time, J. Chin. Inst. Ind. Eng. 24 (2) (2007) 153–165.
[25] R. Bellman, On a routing problem, Q. Appl. Math. 16 (1958) 88–90.
[26] U. Pape, Implementation and efficiency of Moore-algorithms for the shortest route problem, Math. Program. 7 (1974) 212–222.
[27] G. Gallo, S. Pallottino, Shortest path algorithms, Ann. Oper. Res. 7 (1988) 3–79.
[28] F. Glover, D. Klingman, N. Philips, A new polynomial bounded shortest path algorithm, Oper. Res. 33 (1986) 65–73.
[29] D.P. Bertsekas, A simple and fast label correcting algorithm for shortest paths, Networks 23 (1993) 703–709.
[30] R. Kolisch, A. Sprecher, PSPLIB-A project scheduling problem library, European J. Oper. Res. 96 (1996) 205–216.